대한민국특허청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0069946
Application Number

출 원 년 월 일 : 2002년 11월 12일
Date of Application NOV 12, 2002

출 원 인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2003 년 10 월 29 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】 서지사항 보정서
【수신처】 특허청장
【제출일자】 2003.10.23
【제출인】
【명칭】 삼성전자 주식회사
【출원인코드】 1-1998-104271-3
【사건과의 관계】 출원인
【대리인】
【성명】 임창현
【대리인코드】 9-1998-000386-5
【포괄위임등록번호】 1999-007368-2
【대리인】
【성명】 권혁수
【대리인코드】 9-1999-000370-4
【포괄위임등록번호】 1999-056971-6
【사건의 표시】
【출원번호】 10-2002-0069946
【출원일자】 2002.11.12
【심사청구일자】 2002.11.12
【발명의 명칭】 병렬 디이에스 구조를 갖는 암호 장치
【제출원인】
【접수번호】 1-1-2002-0372484-15
【접수일자】 2002.11.12
【보정할 서류】 특허출원서
【보정할 사항】
【보정대상항목】 발명자
【보정방법】 정정
【보정내용】
【발명자】
【성명의 국문표기】 서경덕
【성명의 영문표기】 SEO,KYUNG DUCK
【주민등록번호】 730303-1455618

【우편번호】 449-905
【주소】 경기도 용인시 기흥읍 상갈리 주공아파트 508동 805호
【국적】 KR

【취지】 특허법시행규칙 제13조·실용신안법시행규칙 제8조의 규정에의하여 위와 같 이 제출합니다. 대리인
임창현 (인) 대리인
권혁수 (인)

【수수료】

【보정료】	0	원
【기타 수수료】	원	
【합계】	0	원

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【제출일자】	2002.11.12
【발명의 명칭】	병렬 디이에스 구조를 갖는 암호 장치
【발명의 영문명칭】	CRYPTOGRAPHIC APPARATUS WITH PARALLEL DES STRUCTURE
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	임창현
【대리인코드】	9-1998-000386-5
【포괄위임등록번호】	1999-007368-2
【대리인】	
【성명】	권혁수
【대리인코드】	9-1999-000370-4
【포괄위임등록번호】	1999-056971-6
【발명자】	
【성명의 국문표기】	서경덕
【성명의 영문표기】	SEO,KYOUNG DUCK
【주민등록번호】	730303-1455618
【우편번호】	442-470
【주소】	경기도 수원시 팔달구 영통동 1040-11 105호
【국적】	KR
【심사청구】	청구
【취지】	특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인 임창현 (인) 대리인 권혁수 (인)
【수수료】	
【기본출원료】	20 면 29,000 원
【가산출원료】	6 면 6,000 원

【우선권주장료】	0	건	0 원
【심사청구료】	6	항	301,000 원
【합계】	336,000 원		
【첨부서류】	1. 요약서·명세서(도면)_1통		

【요약서】

【요약】

여기에 개시되는 암호 장치는 제 1 및 제 2의 N-라운드 디이에스 장치들과 제 1 및 제 2 입력 회로들을 포함한다. 제 1의 N-라운드 디이에스 장치는 일련의 암호 키들의 입력에 따라, 디지털 입력 데이터 블록을 제 1 디지털 출력 데이터 블록으로 비선형적으로 암호 변환한다. 제 1 입력 회로는 디지털 입력 데이터 블록을 입력하여 반전시키고, 제 2 입력 회로는 일련의 암호 키들을 입력하여 반전시킨다. 제 2의 N-라운드 디이에스 장치는 반전된 암호 키들의 입력에 따라, 반전된 디지털 입력 데이터 블록을 제 2 디지털 출력 데이터 블록으로 비선형적으로 암호 변환한다. 여기서, 제 1 및 제 2의 N-라운드 디이에스 장치들은 입력되는 대응하는 데이터 블록들 및 키값들에 응답하여 암호 변환 동작을 동시에 수행한다.

【대표도】

도 1

【명세서】

【발명의 명칭】

병렬 디이에스 구조를 갖는 암호 장치{CRYPTOGRAPHIC APPARATUS WITH PARALLEL DES STRUCTURE}

【도면의 간단한 설명】

도 1은 본 발명에 따른 암호 장치를 보여주는 블록도;

도 2는 도 1에 도시된 암호키 블록을 보여주는 블록도;

도 3은 도 1에 도시된 암호 블록들 중 하나를 보여주는 블록도;

도 4는 도 3에 도시된 함수 처리기를 보여주는 블록도; 그리고

도 5는 도 4에 도시된 S 박스들의 변환 스케쥴을 보여주는 도면이다.

* 도면의 주요 부분에 대한 부호 설명 *

100 : 암호 장치

120 : 암호키 블록

140, 160 : 암호 블록

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<10> 본 발명은 데이터 통신 (data communication)에 관한 것으로, 좀 더 구체적으로 디지털 데이터 블록을 암호화/복호화하는 암호 장치에 관한 것이다.

<11> 디지털 입력 블록을 디지털 출력 블록으로 변환하기 위한 폭 넓게 사용된 알고리즘은 연방 정부의 연방 정보 처리 표준 46(FIPS publication 46)으로 채택된 데이터 암호화 표준 알고리즘 (data encryption standard algorithm, 이하 "DES 알고리즘"이라 칭함)이다. 그러한 알고리즘은 일반적으로 블록 암호화기 (block cipher)라 불린다. 암호문 (ciphertext)을 복호화한다는 것은 데이터를 원래의 형태로 변환한다는 것이다. DES는 64-비트 평문 블록들을 대응하는 64-비트 암호문 블록들로 암호화하는 데 사용된다. 이때, 64-비트 키로부터 생성되는 키들을 사용하여 암호화 동작이 수행된다.

<12> 상술한 DES 알고리즘은, 예를 들면, 카드 판독기 (card reader)와 스마트 카드 간의 통신에 사용되고 있다. 잘 알려진 바와 같이, 스마트 카드 내부에 저장된 데이터는 안전하게 보관되어야 하며, 외부로 유출시에는 사용자에게나 시스템 운영자에게도 커다란 위험 인자가 된다. 스마트 카드의 승인되지 않은 접근은 "부정조작" (tampering)이라 불리며, 스마트 카드에 대한 부정조작은 일반적으로 행해지고 있다. 부정조작 기술은 다양한 어택 기술들, 예를 들면, 마이크로프로브 기술 (microprobing technique), 소프트웨어 어택 기술 (software attack technique), 도청 기술 (eavesdropping technique), 그리고 오류 생성 기술 (fault generation technique)로 구분될 수 있다. 스마트 카드를 부정조작함으로써 카드 메모리에 저장된 정보와 적용된 암호 알고리즘의 키값을 얻을 수 있다.

<13> 상기 마이크로프로브 기술은 칩 표면을 직접 액세스하기 위해 사용될 수 있다. 상기 소프트웨어 어택 기술은 프로세서의 일반적인 통신 인터페이스를 이용하며 프로토콜에서 발생되는 보안 취약점 (security vulnerability), 암호 알고리즘,

또는 알고리즘 실행을 활용한다. 도청 기술은 모든 공급 및 인터페이스 접속들의 아날로그 특성들과 정상적인 동작 동안 프로세서에 의해서 생성되는 전자기 방사를 측정한다. 오류 생성 기술은 비정상적인 환경 조건을 이용하여 추가적인 접근을 제공하는 프로세서의 오동작을 생성한다. 상기 마이크로프로브 기술은 직접적인 어택 (invasive attack technique)이며, 이 기술은 많은 시간을 필요로 한다. 나머지 기술들은 간접적인 어택 기술 (non-invasive attack technique)이다.

<14> 상기 간접적인 어택 기술로서, 사이드 채널 분석 (side channel analysis) 기술은 스마트 카드의 동작에 의한 전력 소모 (또는 소모 전류 패턴) 또는 타이밍차 (timing difference)를 이용하여 암호 알고리즘 (또는 DES 알고리즘)의 키값을 알아내는 것을 말한다. 사이드 채널 분석 기술은 크게 SPA (simple power analysis) 기술과 DPA (differential power analysis) 기술로 분류될 수 있다. SPA 기술은 암호 알고리즘이 수행될 때 측정한 전력 자체의 분석을 통해 키값을 추출하는 데 사용된다. DPA 기술은 SPA 개념에 통계적 개념과 오류 수정의 개념을 도입하여 키값을 추출하는 데 사용된다.

<15> DES 알고리즘의 키값과 관련된 데이터가 처리될 때 발생되는 전력 소모 또는 소모 전류 패턴은, 일반적으로, 처리되는 데이터 비트가 "0" 또는 "1"인 지에 따라 미세한 차이를 보인다. 따라서, 미세한 차이를 보이는 소모 전류 패턴들을 정확하게 분류함으로써 데이터 비트 "1"의 소모 전류 패턴과 데이터 비트 "0"의 소모 전류 패턴 사이의 차를 통해 키값을 찾을 수 있다.

<16> 결론적으로, "0" 데이터 비트 및 "1" 데이터 비트의 소모 전류 패턴들 간의 미세한 차가 DPA 기술에 노출되는 것을 방지할 수 있는 향상된 DES 알고리즘이 요구되고 있다.

【발명이 이루고자 하는 기술적 과제】

<17> 본 발명의 목적은 사이드 채널 분석 (side channel analysis)에 강한 암호 장치를 제공하는 것이다.

【발명의 구성 및 작용】

<18> 상술한 제반 목적을 달성하기 위한 본 발명의 암호 장치에 따르면, 제 1의 N-라운드 디이에스 장치는 일련의 암호 키들의 입력에 따라, 디지털 입력 데이터 블록을 제 1 디지털 출력 데이터 블록으로 비선형적으로 암호 변환한다. 제 1 입력 수단은 디지털 입력 데이터 블록을 입력하여 반전시키고, 제 2 입력 수단은 일련의 암호 키들을 입력하여 반전시킨다. 제 2의 N-라운드 디이에스 장치는 반전된 암호 키들의 입력에 따라, 반전된 디지털 입력 데이터 블록을 제 2 디지털 출력 데이터 블록으로 비선형적으로 암호 변환한다. 여기서, 제 1 및 제 2의 N-라운드 디이에스 장치들은 암호 변환 동작을 동시에 수행한다.

<19> 이 실시예에 있어서, 상기 제 1 및 제 2의 N-라운드 디이에스 장치들은 디이에스 알고리즘 (DES algorithm)에 따라 암호 변환 동작을 각각 수행한다.

<20> 이 실시예에 있어서, 상기 제 1 및 제 2의 N-라운드 디이에스 장치들로부터의 상기 제 1 및 제 2 디지털 출력 데이터 블록들을 저장하는 수단을 더 포함하며, 상기 제 1 및 제 2 디지털 출력 데이터 블록들 중 하나만이 암호 데이터 블록으로서 사용된다.

<21> 이 실시예에 있어서, 상기 디지털 입력 데이터 블록을 상기 제 1의 N-라운드 디이에스 장치로 전달하는 제 3 입력 수단을 더 포함한다.

<22> 본 발명의 다른 특징에 따르면, 디지털 입력 데이터를 암호 변환하는 방법은 일련의 암호 키들의 입력에 따라, 상기 디지털 입력 데이터 블록을 제 1 디지털 출력 데이터 블록으로

비선형적으로 암호 변환하는 단계와; 상기 디지털 입력 데이터 블록 및 상기 일련의 암호 키들을 반전시키는 단계와; 그리고 상기 반전된 암호 키들의 입력에 따라, 상기 반전된 디지털 입력 데이터 블록을 제 2 디지털 출력 데이터 블록으로 비선형적으로 암호 변환하는 단계를 포함한다. 여기서, 상기 제 1 및 제 2 디지털 출력 데이터 블록들을 얻기 위한 상기 암호 변환 동작들은 디이에스 알고리즘 (DES algorithm)에 따라 동시에 수행된다. 상기 제 1 및 제 2 디지털 출력 데이터 블록들 중 하나만이 암호 데이터 블록으로서 사용된다.

<23> 이하, 본 발명의 바람직한 실시예에 따른 암호 장치가 참조 도면들에 의거하여 상세히 설명될 것이다. 도 1에는 본 발명에 따른 암호 장치의 블록도가 도시되어 있다. 도 1을 참조하면, 본 발명의 암호 장치 (100)는 64-비트 키에 따라 디지털 입력 데이터 블록 또는 평문 (plaintext)을 암호화하며, 평문은 64-비트 데이터이다. 본 발명의 암호 장치 (100)는 암호키 블록 (encryption key block) (120), 제 1 및 제 2 암호 블록들 (first and second encryption blocks) (140, 160), 레지스터 (180), 버퍼들 (BUF1, BUF2), 그리고 인버터들 (INV1, INV2)을 포함한다.

<24> 도 1에 도시된 바와 같이, 암호키 블록 (120)은 64-비트 키 (KEY)를 받아들이고, 이하 설명될 변환 방식 (permutation method)에 따라 복수 개의 48-비트 키들 (K1-K16)을 발생한다. 그렇게 발생된 16개의 암호키들 (K1-K16)은 버퍼 (BUF1)를 통해 제 1 암호 블록 (140)으로 그리고 인버터 (INV1)를 통해 제 2 암호 블록 (160)으로 각각 전달된다. 이러한 설명으로부터 알 수 있듯이, 제 1 암호 블록 (140)은 암호키 블록 (120)으로부터의 암호키들 (K1-K16)을 그대로 이용하여 암호화 동작을 수행하는 반면에, 제 2 암호 블록 (160)은 암호키 블록 (120)으로부터 출력된 암호키들 (K1-K16)에 대해 1의 보수를 취하여 얻어진 보수 암호키들 (K1'-K16')을 이용하여 암호화 동작을 수행한다. 64-비트 데이터 블록으로서, 디지털 입력 데이터 블록 (D)

은 버퍼 (BUF2)를 통해 제 1 암호 블록 (140)으로 그리고 인버터 (INV2)를 통해 제 2 암호 블록 (160)으로 각각 전달된다. 제 1 암호 블록 (140)은 암호키들 (K1-K16)에 따라 버퍼 (BUF2)로부터의 디지털 입력 데이터 블록 (D)을 암호화하는 반면에, 제 2 암호 블록 (160)은 보수 암호키들 (K1'-K16')에 따라 인버터 (INV2)를 통해 반전된 데이터 블록 (D') (이하, "보수 데이터 블록"이라 칭함)을 암호화한다. 즉, 제 2 암호 블록 (160)은 보수 암호키들 (K1'-K16')에 따라 보수 데이터 블록 (D')을 암호화한다. 암호 블록들 (140, 160)으로부터 출력되는 암호화 데이터 블록들 (C, C')은 레지스터 (180)에 저장되며, 암호화 데이터 블록들 (C, C') 중 하나만이 실질적인 암호화 데이터 블록으로서 사용될 것이다.

<25> 이 실시예에 있어서, 제 1 및 제 2 암호 블록들 (140, 160) 각각은 데이터 암호화 표준 (Data Encryption Standard, 이하 "DES"라 칭함) 알고리즘에 따라 암호화/복호화 동작을 수행하도록 구현되며, "DES 장치"라고도 불린다. 비록 하나의 버퍼 (BUF1) 및 하나의 인버터 (INV1)만이 도 1에 도시되어 있지만, 각 암호키를 구성하는 48개의 데이터 비트들에 각각 대응하는 버퍼들 및 인버터들이 사용됨은 자명하다. 마찬가지로, 비록 하나의 버퍼 (BUF2) 및 하나의 인버터 (INV2)만이 도 1에 도시되어 있지만, 디지털 입력 데이터 블록을 구성하는 64개의 데이터 비트들에 각각 대응하는 버퍼들 및 인버터들이 사용됨은 자명하다.

<26> 앞서의 설명에 따르면, 본 발명에 따른 암호 장치 (100)는 DES 알고리즘을 이용하여 각 디지털 입력 데이터 블록을 암호화/복호화하도록 (enciper/decipher) 설계된다. DES 알고리즘을 이용한 암호 장치는 64-비트 키 (또는 암호키)에 따라 64-비트 데이터를 암호화한다. 복호화 (deciphering)는 암호화하는 데 사용된 것과 동일한 키를 이용함으로써 달성될 수 있다. 특히, 본 발명의 암호 장치 (100)는, 도 1에 도시된 바와 같이, 디지털 입력 데이터 블록 (또는 평문)을 개별적으로 그리고 동시에 암호화하는 2개의 암호 블록들 (140, 160) (또는 DES 장치

들)을 갖는다. 암호 블록들 중 하나 (140)는 암호키들 (K1-K16) 및 데이터 블록 (D)을 그대로 이용하여 암호화 동작을 수행하는 반면에, 다른 하나 (160)는 보수 암호키 (K1'-K16') 및 보수 데이터 블록 (D')을 이용하여 암호화 동작을 수행한다. 이는 하나의 암호화 블록에서 '1'의 데이터 비트가 처리될 때 다른 암호화 블록에서 '0'의 데이터 비트가 처리됨을 의미한다. 이러한 병렬 암호화 방식에 따르면, 데이터 블록을 암호화할 때 생기는 전류 패턴 (current pattern)을 이용하여 키값을 찾는 것이 어렵다.

<27> 도 2는 도 1에 도시된 암호키 블록의 바람직한 실시예를 보여주는 블록도이다. 키 (KEY)는 64개의 비트들로 구성되며, 64개의 비트들 중 56개의 비트들만이 알고리즘에 사용된다. 64-비트 키 (KEY)는 PC1 박스를 통해 54-비트 키 (K+)로 변환된다(permuted). PC1 박스의 변환표는 아래와 같다.

<28> 【표 1】

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

<29> 표 1의 첫 번째 번지 (first entry)가 "57"이기 때문에, 원시키 (original key) (KEY)의 57번째 비트가 변환된 키 (K+)의 첫 번째 비트가 됨을 의미한다. 원시키 (KEY)의 49번째 비트는 변환된 키 (K+)의 두 번째 키가 된다. 원시키 (KEY)의 네 번째 비트는 변환된 키 (K+)의 마지막 비트가 된다. 원시키의 64개의 비트들 중 54개의 비트들이 변환된 키 (K+)로 나타난다. 예를 들면; "00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001"인

64-비트 키 (KEY)로부터 "1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111"인 56-비트 변환키 (K+)를 얻을 수 있다.

<30> 그 다음에, 변환키 (K+)는 각각 28개의 비트들로 구성되는 2개의 왼쪽 및 오른쪽 블록들 (C0, D0)로 분리된다. 예를 들면, 변환키 (K+)로부터, C0 = 1111000 0110011 0010101 0101111와 D0 = 0101010 1011001 1001111 0001111을 얻을 수 있다. 아래의 표 2에 따라 이전 블록의 비트들을 좌측으로 쉬프트시킴으로써 16개의 블록 쌍들 (Cn, Dn) ($1 \leq n \leq 16$)을 얻을 수 있다. 좌측으로 쉬프트하기 위해서, 첫 번째 비트를 제외하고, 각 비트는 두 번째 비트가 블록의 끝으로 회전되도록 좌측으로 한자리씩 이동된다.

<31> 【표 2】

회전위치	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
쉬프트횟수	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

<32> 예를 들면, C3와 D3는 2번의 좌측 쉬프트 (two left shifts)를 통해 C2와 D2로부터 각각 얻어지고, C16과 D16은 1번의 좌측 쉬프트를 통해 C15와 D15로부터 각각 얻어진다.

<33> 마지막으로, 아래의 표 3에 따라 16개의 PC2 박스들을 통해 16개의 키들 (K1-K16)이 생성될 것이다. PC2 박스의 변환표는 아래와 같다.

<34>

【표 3】

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

<35> 첫 번째 키 (K1)는 표 3의 변환 스케쥴에 의거하여 결합된 형태의 블록 (C1D1)으로부터 얻어지고, 마지막 키 (K16)는 표 3의 변환 스케쥴에 의거하여 결합된 형태의 블록 (C16D16)으로부터 얻어진다. 예를 들면, PC2 박스에 C1D1 블록을 인가함으로써, K1 = 000110 110000 001011 101111 111111 000111 000001 110010 (48-비트 키)이 된다. 이러한 방식을 통해 각 블록 (C2D2)-(C16D16)으로부터 각 키 (K2-K16)이 얻어질 것이다.

<36> 앞서 설명에 따라 생성된 16개의 48-비트 키들 (K1-K16)은 버퍼 (BUF1)를 통해 제 1 암호 블록 (140)으로 그리고 인버터 (INV1)를 통해 제 2 암호 블록 (160)으로 각각 전달된다.

<37> 도 3은 도 1에 도시된 제 1 및 제 2 암호 블록들 중 하나를 보여주는 블록도이고, 도 4는 도 3에 도시된 함수처리기를 보여주는 블록도이다. 도 3에는 단지 하나의 암호 블록 (예를 들면, 140)만이 도시되어 있지만, 나머지 역시 도 3에 도시된 것과 동일하게 구성된다. 암호 블록 (140)은 초기 순열기 (141), 최종 순열기 (142), 그리고 복수 개의, 예를 들면, 16개의 라운드들로 구성되며, 각 라운드는 함수 처리기 (f)와 XOR 연산기 (+)로 구성된다.

<38> 먼저 도 3을 참조하면, 64-비트 평문 (D)은 도 1에 도시된 버퍼 (BUF2)를 통해 전달되며, 그것의 비트 순서가 초기 순열기 (initial permutation unit) (141)를 통해 변환된다. 즉, 평문의 비트들은 아래의 표 4를 통해 재배열된다. 표 4에서 알 수 있듯이, 평문 (D)의 58번째

비트는 변환된 평문 (IP)의 첫 번째 비트가 된다. 평문 (D)의 50번째 비트는 변환된 평문 (IP)의 두 번째 비트가 된다. 평문 (D)의 7번째 비트는 변환된 평문 (IP)의 마지막 비트가 된다.

<39> 【표 4】

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

<40> 표 4의 변환 스케쥴을 평문 (D = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111)에 적용하면 변환된 블록 (IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 00001 1010 1010)을 얻을 수 있다.

<41> 여기서, 평문 (D)의 58번째 비트는 변환된 블록 (IP)의 첫 번째 비트가 된다. 평문 (D)의 50번째 비트는 "1"이며 이는 변환된 블록 (IP)의 두 번째 비트가 된다. 평문 (D)의 7번째 비트는 "0"이며 이는 변환된 블록 (IP)의 마지막 비트가 된다.

<42> 그 다음에, 변환된 블록 (IP)은 32개의 비트들로 구성되는 왼쪽 및 오른쪽 블록들 (L0, R0)로 분리된다. 예를 들면, 변환된 블록 (IP)로부터, L0 = 1100 1100 0000 0000 1100 1100 1111 1111과 R0 = 1111 0000 1010 1010 1111 00001 1010 1010을 얻을 수 있다. 기호 "+"가 XOR 가산 (XOR addition) (또는 bit-by-bit addition modulo 2)을 나타낸다고 가정하자. 마지막 라운드의 경우, $L_n = R_{n-1}$이 되고, $R_n = L_{n-1} + f(R_{n-1}, K_n)$이 된다. 즉, 각 라운드에 있어서, 이전 결과의 우측의 32개의 비트들은 현 라운드에서 좌측의 32개의 비트들이 된다. 현 라운드에

서 우측의 32개의 비트들의 경우, 이전 라운드의 좌측의 32개의 비트들을 함수 처리기 (f)의 결과와 함께 XOR 연산한다.

<43> 예를 들면, n=1인 경우,

<44> K1 = 000110 110000 001011 101111 111111 000111 000001 110010

<45> L1 = R0 = 1111 0000 1010 1010 1111 0000 1010 1010

<46> R1 = Lo + f(Ro, K1)이라 가정하자.

<47> 함수 처리기 (f)는, 도 4에 도시된 바와 같이, 각 블록 (Rn-1)을 32개의 비트들에서 48개의 비트들로 확장한다. 이는 Rn-1 블록의 비트들 중 일부를 반복시키는 선택 테이블 (표 5 참조)을 이용하여 행해진다. 선택 테이블이 사용되는 경우, 선택 테이블의 사용을 함수 (E)로 표기하기로 가정하자. 따라서 E(Rn-1)은 32-비트 입력 블록과 48-비트 출력 블록을 갖는다. 출력 블록의 48개의 비트들은 아래의 표 5에 따라 입력 블록의 32개의 비트들을 선택함으로써 얻어진다.

<48> 【표 5】

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

<49> 예를 들면, E(R0)는 다음과 같이 R0 블록으로부터 산출된다.

<50> R0 = 1111 0000 1010 1010 1111 0000 1010

<51> E(R0) = 011110 100001 010101 010101 011110 100001 010101 010101

<52> 즉, 4개의 원시 비트들로 구성된 각 블록은 6개의 출력 비트들로 구성된 블록으로 확장된다.

<53> 함수 처리기 (f)에 있어서, 도 4에 도시된 바와 같이, 출력 블록 (E(Rn-1))은 키 (Kn)와 XOR 연산된다. 연산 결과는 Kn + E(Rn-1)로 표현된다. 예를 들면, K1 = 000110 110000 001011 101111 111111 000111 000001 110010이고 E(R0) = 011110 100001 010101 010101 011110 100001 010101 010101인 경우, Kn + E(Rn-1) = 011000 010001 011110 111010 100001 100110 010100 100111이 된다.

<54> XOR 연산된 결과 (Kn + E(Rn-1))로서, 48개의 비트들은 각각 6개의 비트들로 구성되는 8개의 그룹들로 분리된다. 각 그룹의 비트들은 "S 박스"라 불리는 대응하는 테이블에서 어드레스로서 사용된다. 대응하는 S 박스들 (S1-S8)은 6-비트 입력 블록들을 4-비트 출력 블록들로 각각 변환된다. 최종적으로 32 비트들이 얻어진다. 이전의 결과는 아래의 형식으로 표현될 수 있다.

<55> Kn + E(Rn-1)=B1B2B3B4B5B6B7B8

<56> 여기서, 각 Bj (j=1-8)는 6개의 비트들로 구성된 그룹이다. S 박스들 (S1-S8)을 통해 출력되는 결과는 S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8)으로 표현될 수 있다. 각 함수들 (S1, S2, …, S8)은 입력으로서 6-비트 블록을 받아들여 출력으로서 4-비트 블록을 출력한다. 예를 들면, S1 박스의 변환 스케쥴은 아래의 표 6과 같다.

<57>

【표 6】

	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
R0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
R1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
R2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
R3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

<58> 여기서, "R"은 행을 나타내고 "C"는 열을 나타낸다.

<59> 만약 S1이 표 6에서 정의된 함수이고 B이 6-비트 블록이면, S1(B)는 다음과 같이 결정된다. B의 첫 번째 및 마지막 비트들은 0에서 3까지의 10진수 범위 내의 2진수를 나타낸다. 그 수를 i라고 하자. B의 나머지 4개의 비트들은 0에서 16까지의 10진수 범위 내의 2진수를 나타낸다. 그 수를 j라고 하자. 이러한 원리에 따라, i번째 행 및 j번째 열의 수가 선택된다. 선택된 수는 4-비트 블록이 된다. 예를 들면, 입력 블록 (B)이 "011011"인 경우, 첫 번째 비트는 "0"이고 마지막 비트는 "1"이 되며, 그 결과 행 어드레스로서 "01"이 주어진다. 이는 두 번째 행 (R1)이 선택됨을 의미한다. 나머지 4개의 비트들은 "1101"이며, 이는 10진수 13을 나타내는 이진수이다. 즉, 14번째 열 (C13)이 선택된다. 표 6에서 알 수 있듯이, R1과 C13에 위치한 수는 5이며, 5는 이진수 0101이다. 그러므로 S1(011011)=0101. 나머지 S 박스들의 변환표가 도 5에 도시되어 있다. 나머지 S 박스들 역시 앞서 설명된 것과 동일한 방식으로 6-비트 블록을 4-비트 블록으로 변환한다.

<60> 예를 들면, 첫 번째 라운드의 경우, 8개의 S 박스들의 출력으로서 아래의 결과를 얻을 수 있다.

<61> K1 + E(R0) = 011000 010001 011110 111010 100001 100110 010100 100111

<62> S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111

<63> 함수 처리기 (f)의 마지막 단계는 S 박스들의 출력을 변환 (permutation: P)하는 것이다. 함수 처리기 (f)의 최종적인 값은 f = P(S1(B1)S2(B2)...S8(B8))이 된다. 변환 (P)은 아래의 표 7에 의해서 정의된다. 입력 블록의 비트들을 재배열함으로써 32-비트 입력에서 32-비트 출력이 생성된다.

<64> 【표 7】

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

<65> 예를 들면, 8개의 S 박스들의 출력

<66> S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111으로부터

<67> 최종적인 값 (f) "0010 0011 0100 1010 1010 1001 1011 1011"을 얻을 수 있다.

<68> R1 = L0 + f(R0, K1)

<69> = 1100 1100 0000 0000 1100 1100 1111 1111

<70> + 0010 0011 0100 1010 1010 1001 1011 1011

<71> = 1110 1111 0100 1010 0110 0101 0100 0100

<72> 다시 도 3을 참조하면, 다음의 라운드에서, L2는 앞서의 계산으로 얻어진 R1이 되고 R2는 L1 + f(R0, K1)이 된다. 나머지 라운드들 역시 앞서 설명된 것과 동일한 방법으로 동작한다. 마지막 라운드의 출력으로서, L16 및 R16 블록들이 생성된다. 2개의 블록들의 순서는 R16L16의 64-비트 블록이 되도록 뒤집어진다. 그렇게 얻어진 64-비트 블록의 비트 순서는 아래의 표 8에 따라 최종 순열기 (142)를 통해 재배열된다.

<73> 【표 8】

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	45	11	51	19	59	27
34	2	43	10	50	18	58	26
33	1	41	9	49	17	57	25

<74> 앞서 설명된 방법을 사용하여 16개의 라운드들을 처리한 경우,

<75> L16 = 0100 0011 0100 0010 0011 0010 0011 0100

<76> R16 = 0000 1010 0100 1100 1101 1001 1001 0101이라고 가정하자.

<77> 2개의 블록들의 순서를 바꾸고 최종 순열기 (142)에 R16L16 블록을 인가하면,

<78> R16L16 = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

<79> IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101이 된다.

<80> 이는 16진수 형태로 "85E813540F0AB405"이 된다. 결론적으로 평문 (M=0123456789ABCDEF)는 암호문 (C=85E813540F0AB405)으로 암호화된다.

<81> 앞서 설명된 바와 같이, 본 발명에 따른 암호 장치는 2개의 암호 블록들 (140, 160)을 포함한다. 암호 블록들 (140, 160)은 앞서 설명된 것과 동일한 방식으로 암호화 동작을 수행한다. 특히, 암호 블록 (140)은 평문 (D)와 암호키들 (K1-K16)을 그대로 사용하는 반면에, 암호 블록 (160)은 보수 평문 (D')와 보수 암호키들 (K1'-K16')을 사용한다. 일반적으로, 함수 처리기 (f)가 동작할 때 가장 많은 전류가 소모되기 때문에, "0" 비트를 처리할 때 생기는 소모 전류 패턴은 "1" 비트를 처리할 때 생기는 소모 전류 패턴과 다르다. 그러한 까닭에, 소모 전류 패턴을 모니터링(또는 분석)함으로써 암호화 과정에 사용되는 키값을 찾는 것이 가능하다. 본 발명의 경우, 하지만, 제 1 암호 블록 (140)의 라운드의 함수 처리기 (f)에서 "0" 비트가 처리될 때, 제 2 암호 블록 (160)의 대응하는 라운드의 함수 처리기 (f)에서 "1" 비트가 처리된다. 즉, 암호 블록들 (140, 160)의 대응하는 함수 처리기들이 서로 상반된 데이터 값들을 처리하기 때문에, "0" 및 "1" 비트들을 처리할 때 각각 생기는 소모 전류 패턴들 간의 차이는 현저히 감소될 수 있다.

<82> 이상에서, 본 발명에 따른 회로의 구성 및 동작을 상기한 설명 및 도면에 따라 도시하였지만, 이는 예를 들어 설명한 것에 불과하며 본 발명의 기술적 사상 및 범위를 벗어나지 않는 범위 내에서 다양한 변화 및 변경이 가능함은 물론이다.

【발명의 효과】

<83> 상술한 바와 같이, 암호 블록들 (140, 160)의 대응하는 함수 처리기들이 서로 상반된 데이터 값들을 동시에 처리하기 때문에, "0" 및 "1" 비트들을 처리할 때 각각 생기는 소모 전류 패턴들 간의 차이는 현저히 감소될 수 있다. 그러므로, 소모 전류 패턴을 이용하여 키값을 찾는 것이 어렵다.

【특허청구범위】

【청구항 1】

일련의 암호 키들의 입력에 따라, 디지털 입력 데이터 블록을 제 1 디지털 출력 데이터 블록으로 비선형적으로 암호 변환하는 제 1의 N-라운드 디이에스(DES) 장치와;

상기 디지털 입력 데이터 블록을 입력하여 반전시키는 제 1 입력 수단과;

상기 일련의 암호 키들을 입력하여 반전시키는 제 2 입력 수단과; 그리고

상기 반전된 암호 키들의 입력에 따라, 상기 반전된 디지털 입력 데이터 블록을 제 2 디지털 출력 데이터 블록으로 비선형적으로 암호 변환하는 제 2의 N-라운드 디이에스 장치를 포함하며, 상기 제 1 및 제 2의 N-라운드 디이에스 장치들은 암호 변환 동작을 동시에 수행하는 것을 특징으로 하는 암호 장치.

【청구항 2】

제 1 항에 있어서,

상기 제 1 및 제 2의 N-라운드 디이에스 장치들은 디이에스 알고리즘 (DES algorithm)에 따라 암호 변환 동작을 각각 수행하는 것을 특징으로 하는 암호 장치.

【청구항 3】

제 1 항에 있어서,

상기 제 1 및 제 2의 N-라운드 디이에스 장치들로부터의 상기 제 1 및 제 2 디지털 출력 데이터 블록들을 저장하는 수단을 더 포함하며, 상기 제 1 및 제 2 디지털 출력 데이터 블록들 중 하나만이 암호 데이터 블록으로서 사용되는 것을 특징으로 하는 암호 장치.

【청구항 4】

제 1 항에 있어서,

상기 디지털 입력 데이터 블록을 상기 제 1의 N-라운드 디이에스 장치로 전달하는 제 3 입력 수단을 더 포함하는 것을 특징으로 하는 암호 장치.

【청구항 5】

디지털 입력 데이터를 암호 변환하는 방법에 있어서:

일련의 암호 키들의 입력에 따라, 상기 디지털 입력 데이터 블록을 제 1 디지털 출력 데이터 블록으로 비선형적으로 암호 변환하는 단계와;

상기 디지털 입력 데이터 블록 및 상기 일련의 암호 키들을 반전시키는 단계와; 그리고

상기 반전된 암호 키들의 입력에 따라, 상기 반전된 디지털 입력 데이터 블록을 제 2 디지털 출력 데이터 블록으로 비선형적으로 암호 변환하는 단계를 포함하며, 상기 제 1 및 제 2 디지털 출력 데이터 블록들을 얻기 위한 상기 암호 변환 동작들은 디이에스 알고리즘 (DES algorithm)에 따라 동시에 수행되는 것을 특징으로 하는 방법.

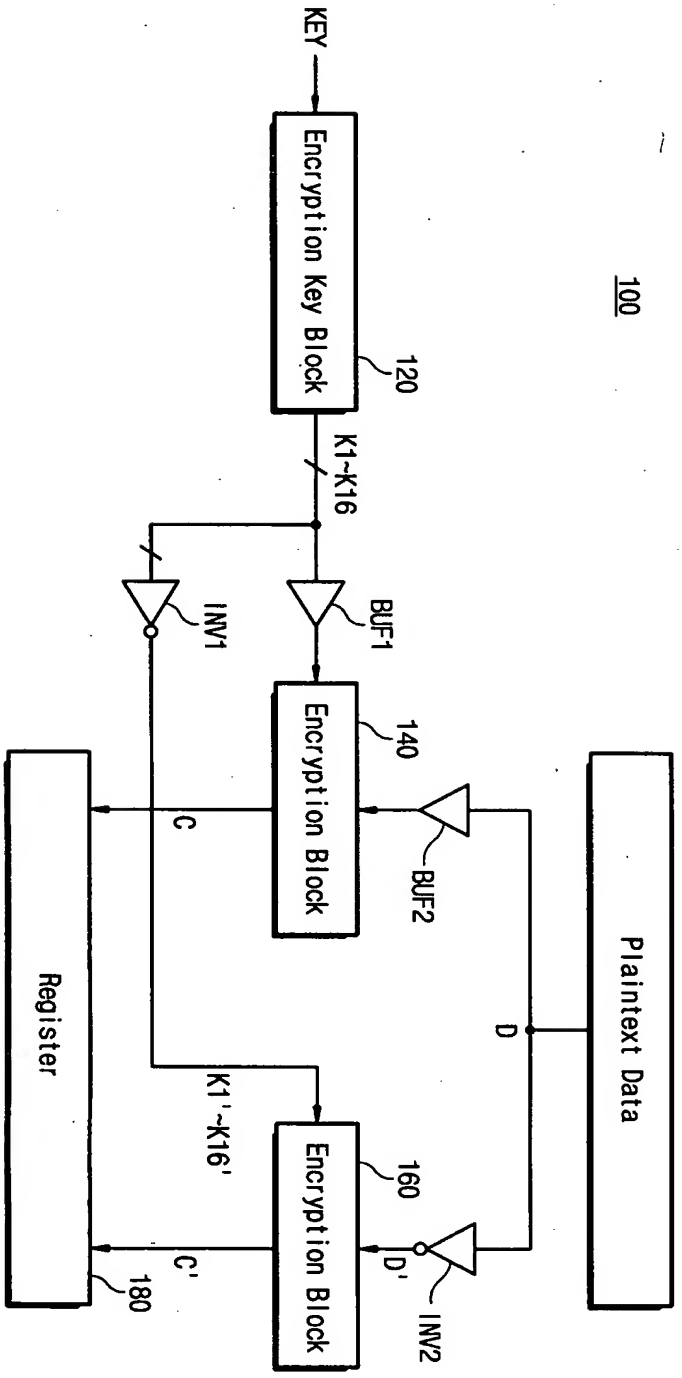
【청구항 6】

제 5 항에 있어서,

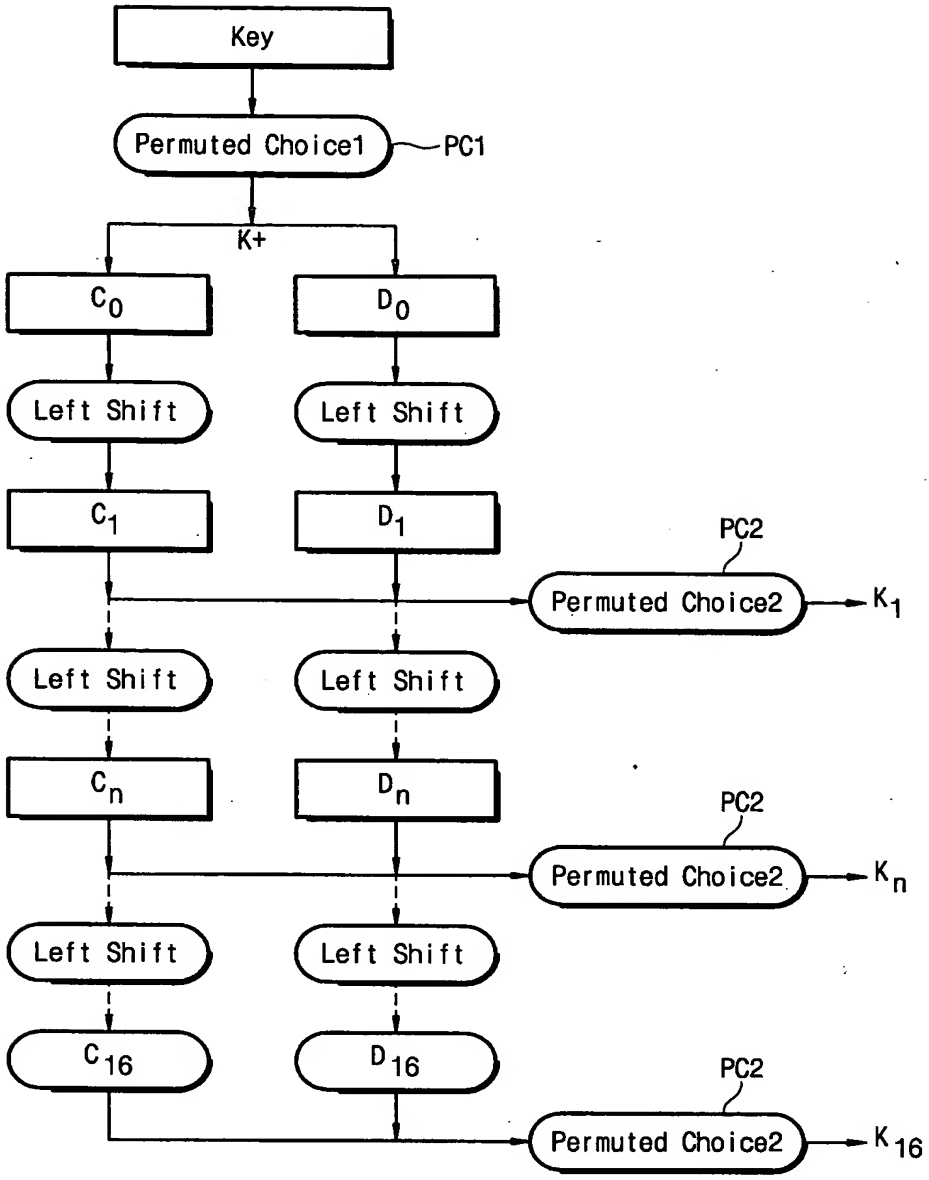
상기 제 1 및 제 2 디지털 출력 데이터 블록들 중 하나만이 암호 데이터 블록으로서 사용되는 것을 특징으로 하는 방법.

【도면】

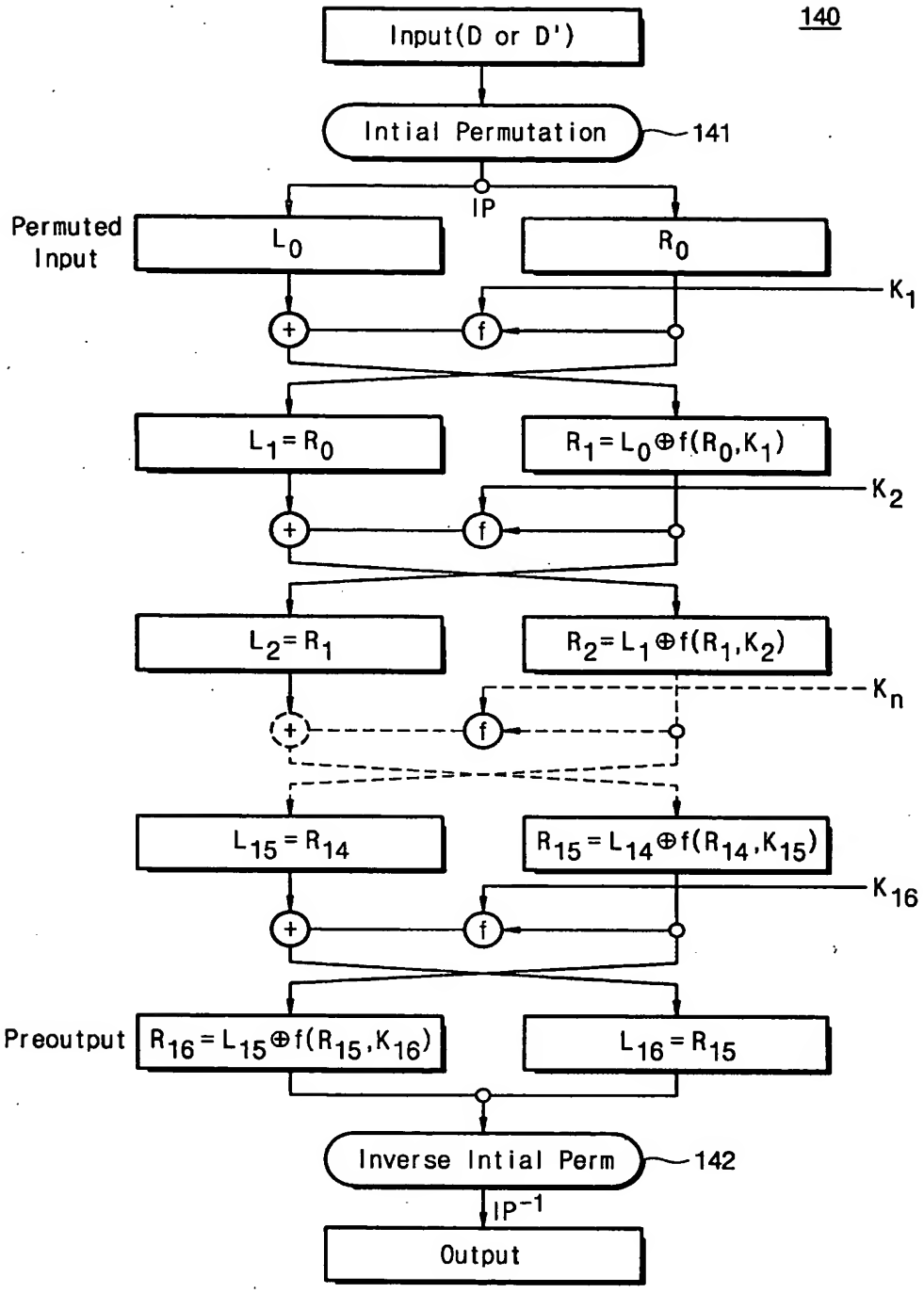
【도 1】


100
KEY
Encryption Key Block
120
K1~K16
INV1
BUF1
Encryption Block
140
C
BUF2
Register
Plaintext Data
D
K1'~K16'
Encryption Block
160
C'
D'
180
INV2

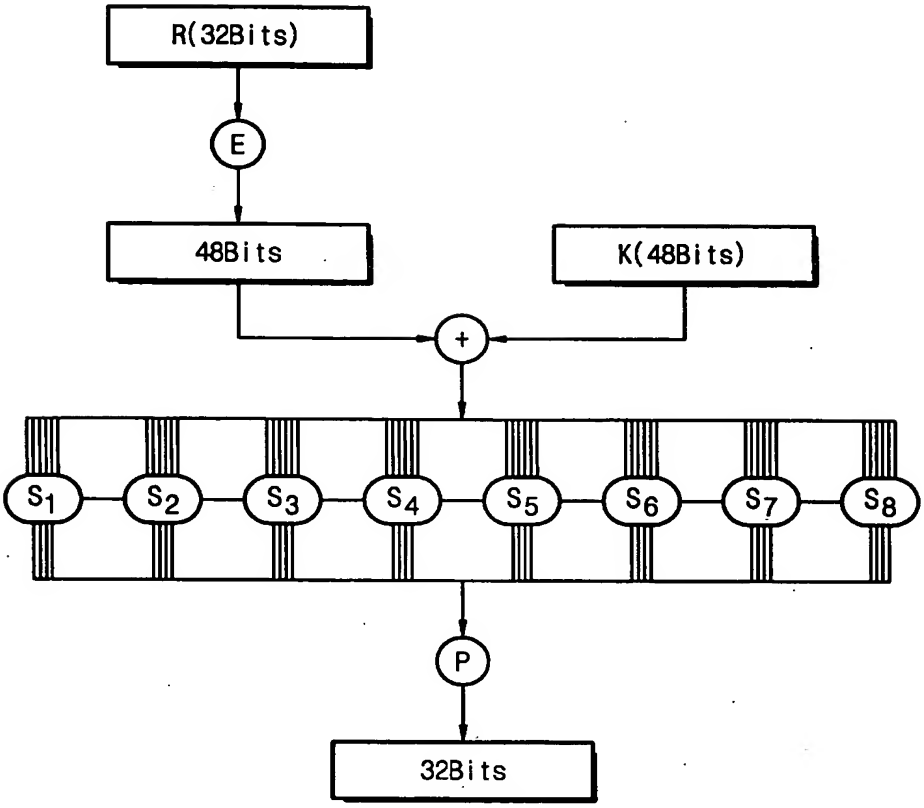
【도 2】


Key
Permuted Choice1
PC1
K+
C0
D0
Left Shift
Left Shift
C1
D1
PC2
Permuted Choice2
K1
Left Shift
Left Shift
Cn
Dn
PC2
Permuted Choice2
Kn
Left Shift
Left Shift
C16
D16
PC2
Permuted Choice2
K16

【도 3】


140
Input(D or D')
Intial Permutation
141
IP
Permuted Input
L_0
R_0
K_1
$L_1=R_0$
$R_1=L_0 \oplus f(R_0,K_1)$
K_2
$L_2=R_1$
$R_2=L_1 \oplus f(R_1,K_2)$
K_n
$L_{15}=R_{14}$
$R_{15}=L_{14} \oplus f(R_{14},K_{15})$
K_{16}
Preoutput
$R_{16}=L_{15} \oplus f(R_{15},K_{16})$
$L_{16}=R_{15}$
Inverse Intial Perm
142
IP^{-1}
Output

【도 4】



【도 5】

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11